2003-09-28

XML binary serialization using cross-fomat schema protocol (XFSP) and XML compression consideration for extensible 3D (X3D) graphics

Brutzman, Don

Naval Postgraduate School (U.S.)

XML Binary Serialization using Cross-Format Schema Protocol (XFSP) and XML Compression Considerations for Extensible 3D (X3D) Graphics

Don Brutzman and Don McGregor, Naval Postgraduate School, Monterey California USA
Alan Hudson, Yumetech Inc., Seattle Washington USA

28 September 2003

1. What work has your organization done in this area? (We are particularly interested in measurements!)

The NPS Cross-Format Schema Protocol (XFSP) has been developed as a general approach to binary serialization of XML documents. Elements and attributes are replaced via a tokenization scheme which carefully preserves valid XML document structure. XFSP uses XML schema as the basis for determining key document parameters such as legal elements, attributes and data types. Originally motivated by the flexible definition of networking protocols, binary serialization of XML via XFSP appears suitable both for message streams and document-storage streams.

An open-source XFSP software implementation written in Java demonstrates the viability of the XFSP approach for XML document serialization and deserialization. Compression measurements using Virtual Reality Modeling Language (VRML 97) scenes have shown expected compression results:

- XML encoding of non-XML data (e.g. translation of VRML .wrl scenes into .x3d documents) typically increases file size
- XML-aware compression of XML encoded data (in this case via XFSP) provides superior compression to gzip compression of the original data. This occurs because the regularity of document structure is more exploitable than alphanumeric patterns.
- Deserialization of compressed XML as in-memory data structures provides higher performance since a second parse of string-based data is not required.

Looking ahead, the Web3D Consortium's X3D Specification Team has released a Request for Proposals (RFP) for an Extensible 3D (X3D) Graphics Compressed Binary Encoding. This effort expects to compose geometric compression techniques with XML-capable binary serialization. This X3D encoding intends to preserve additional markup such as XML signature, XML encryption and other metadata. Thus we expect that W3C normalization of XML binary compression techniques might have significant value, for X3D and other XML-based languages.

The attached white paper and various references provide significant additional detail on both XFSP and X3D Compressed Binary Encoding. (Please note that XFSP is not guaranteed to become part of X3D, but it has been offered for consideration as a baseline framework.)

2. What goals do you believe are most important in this area? (e.g. reducing bandwidth usage; reducing parse time; simplifying APIs or data structures, or other goals)

Binary file size, reduced transmission time and reduced parsing/loading time are all important priorities for this work. The X3D Binary Encoding Request for Proposals (RFP) lists 10 considerations, all of which appear to be composable. They are excerpted as follows. The interoperability requirements for X3D (points 1 and 2) establish the context for these requirements. Nevertheless X3D requirements are not unusual. Thus we expect that all of these points might suitably apply (in varying degrees) to most other XML-based languages.

1. **X3D Compatibility**. The compressed binary encoding shall be able to encode all of the abstract functionality described in X3D Abstract Specification.
2. **Interoperability**. The compressed binary encoding shall contain identical information to the other X3D encodings (XML and Classic VRML). It shall support an identical round-trip conversion between the X3D encodings.
3. **Multiple, separable data types.** The compressed binary encoding shall support multiple, separable media data types, including all node (element) and field (attribute) types in X3D. In particular, it shall include geometric compression for the following.
    - **Geometry** - polygons and surfaces, including NURBS
    - **Interpolation data** - spline and animation data, including particularly long sequences such as motion capture (also see Streaming requirement)
    - **Textures** - PixelTexture, other texture and multitexture formats (also see Bundling requirement)
    - **Array Datatypes** - arrays of generic and geometric data types
    - **Tokens** - tags, element and attribute descriptors, or field and node textual headers
4. **Processing Performance**. The compressed binary encoding shall be easy and efficient to process in a runtime environment. Outputs must include directly typed scene-graph data structures, not just strings which might then need another parsing pass. End-to-end processing performance for construction of a scene-graph as in-memory typed data structures (i.e. decompression and deserialization) shall be superior to that offered by gzip and string parsing.
5. **Ease of Implementation**. Binary compression algorithms shall be easy to implement, as demonstrated by the ongoing Web3D requirement for multiple implementations. Two (or more) implementations are needed for eventual advancement, including at least one open-source implementation.
6. **Streaming**. Compressed binary encoding will operate in a variety of network-streaming environments, including http and sockets, at various (high and low) bandwidths. Local file retrieval of such files shall remain feasible and practical.
7. **Authorability**. Compressed binary encoding shall consist of implementable compression and decompression algorithms that may be used during scene-authoring preparation, network delivery and run-time viewing.
8. **Compression**. Compressed binary encoding algorithms will together enable effective compression of diverse datatypes. At a minimum, such algorithms shall support lossless compression. Lossy compression alternatives may also be supported. When compression results are claimed by proposal submitters, both lossless and lossy characteristics must be described and quantified.
9. **Security**. Compressed binary encoding will optionally enable security, content protection, privacy preferences and metadata such as encryption, conditional access, and watermarking. Default solutions are those defined by the W3C Recommendations for XML Encryption and XML Signature.
10. **Bundling**. Mechanisms for bundling multiple files (e.g. X3D scene, Inlined subscenes, image files, audio file, etc.) into a single archive file will be considered.
11. **Intellectual Property Rights (IPR)**. All technology submissions must follow the predeclaration requirements of the Web3D Consortium IPR policy in order to be considered for inclusion.

X3D Compressed Binary Encoding Requirements

3. **What sort of documents have you studied the most? (e.g. gigabyte-long relational database table dumps; 20-MByte telephone exchange repair manuals; 2 KByte web-service requests)**

   A variety of document sizes and purposes are expected.

   - Short messages, such as agent-to-agent invocations, chat or web service requests
   - Moderate documents, such as typical 3D scenes (5..100 KB)
   - Large static scenes, such as compressed terrain
   - Large streams, such as motion-capture data (positions and orientations for joints and limb segments)

4. **What sorts of applications did you have in mind?**

We are great fans of text-based XML encodings. Nevertheless a large variety of applications might benefit from the availability of compatible alternatives that preserve the XML Infoset capabilities of XML documents within a binary encoding.

- Bandwidth-sensitive applications
- Applications that are slow to execute due to parsing large data documents
- Noisy network links, such as radio or acoustic communications, which can benefit from imposition of forward error correction (FEC) of compressed binary data

5. **If you implemented something, how did you ensure that internationalization and accessibility were not compromised?**

We have not done yet performed specific testing of internationalization (I181N) and Web Accessibility Initiative (WAI) requirements. Nevertheless, XFSP appears suitable for both due to the following reasons:

- Compressed form preserves XML document structure and validability.
- Full support provided for element and attribute information, as defined in XML Schema.
- Given multiple schemas of interest, namespace awareness and multiple-namespace support can be added to the serialization/deserialization algorithms without difficulty. Namespace-support implementation is planned as upcoming work.
- Datatype-specific compression algorithms for Unicode character data payloads are available (e.g. gzip). If further optimization is desired, specialty compression algorithms might conceivably be substituted using similar extensibility mechanisms as defined in XML Signature and XML Encryption.

6. How does your proposal differ from using gzip on raw XML?

Gzip primarily exploits alphanumeric character patterns to compress/decompress strings.

XFSP takes advantage of XML document structure by tokenizing elements and attributes, providing greater optimizability. XFSP then takes advantage of datatype-specific compression possibilities. One simple example of compression advantage is conversion of long floating-point or double-precision alphanumeric strings into IEEE floats and doubles. Thus a further XFSP advantage is direct creation of in-memory data structures, eliminating the gzip requirement for further string-based parsing.

7. Does your solution work with any XML? How is it affected by choice of Schema language? (e.g. W3C XML Schema, DTD, Relax NG)

The XFSP approach can work with any well-defined XML tagset.

- XFSP utilizes W3C XML Schema definitions directly to determine the compression scheme.
- Extending XFSP to DTDs is a simple matter. However payload datatypes would all have to be treated as strings, leaving element/attribute tokenization as the primary benefit. In practice we expect to produce type-aware schemas corresponding to legacy DTDs whenever such binary compression is desired.
- We have not yet attempted to apply XFSP to Relax NG.

8. How important to you are random access within a document, dynamic update and streaming, and how do you see a binary format as impacting these issues?

There are a number of streaming-related issues. In general, it seems best to consider binary XML compressed documents as streams, best suited for network or file-system streaming. Seen from this perspective, binary-XML streams are a likely basis for networking protocols.

- Random access within a document remains important, but does not necessarily have to be performed within the binary format itself. Certainly IDs must be preserved and valid XPath expressions ought to remain consistent. Anything less than preservation of XML Infoset information needs to be avoided, except perhaps as an author-selectable option. Random access can be performed within an XFSP-compressed document. Random access with modification (deletion, addition or some other dynamic update) can also be implemented, but might not be worth specifying since such operations are better performed using in-memory data structures.
- Dynamic update of a document within the confines of a specific binary encoding does not seem to be of particular importance. Based on long experience with a large number of VRML/X3D run-time engines, each of which critically depends on system performance, deserialization to create application-specific in-memory data structures is much more desirable than creation of "standard" data structures. Thus application demand for a "Binary Document Object Model (BDOM)" seems unlikely to occur soon.

- Dynamic update of a protocol itself provides interesting new challenges. Two of the original motivations for NPS research into XFSP were to simplify the creation of customized network protocols, and to permit run-time optimization of such protocols. Schema-based protocol generation can be implemented in several ways: run-time creation of data-binding objects, or autogeneration and autocompilation of corresponding source code. In either approach, using a componentization approach permits run-time loading of new protocol objects. Thus such flexibility is possible, even though it is not of much practical use for stably defined tagsets. We expect such a run-time replacement capability to be useful when optimizing protocols during extended real-world testing. Conceivably such flexibility can also be applied in "always on" virtual worlds where new entities with new behaviors (and corresponding new protocols) are being created and then joining existing shared worlds.
- Streaming in X3D is considered important for several cases. Streaming audio and video are both key capabilities in multimedia scenes. X3D connects such streams via url fields or customized Script nodes. 3D-specific animation is also a streaming goal, for example the ongoing transport of prerecorded (or live, or autogenerated) motion-capture data. Certain geometric algorithms such as continuous mesh refinement or continuous level of detail (CLOD) have also been demonstrated in commercial products and the academic literature. Synchronization of diverse multimedia streams via Synchronized Multimedia Integration Language (SMIL). Such capabilities have not yet been decided upon for X3D, and are candidates for consideration under the X3D Compressed Binary Encoding RFP process.

9. **References**.

a. [Serin 2003] Serin, Ekrem, *Design and Test of the Cross-Format Schema Protocol (XFSP) for Networked Virtual Environments*, Master's Thesis, Naval Postgraduate School, Monterey California USA, March 2003. Thesis available at http://theses.nps.navy.mil/03Mar_Serin.pdf, abstract at http://www.web3d.org/WorkingGroups/x3d-contributors/hypermail/2003/0243.html

b. [X3D RFP 2003] X3D Specification Team, *X3D Compressed Binary Encoding Request For Proposals (RFP)*, Web3D Consortium, July 29 2003. Available at http://www.web3d.org/TaskGroups/x3d/X3dBinaryRFP.html

c. Call for Participation, *W3C Workshop on Binary Interchange of XML Information Item Sets*. http://www.w3.org/2003/07/binary-xml-cfp.html

d. XFSP sample implementation, http://sourceforge.net/projects/npsnetv

10. **Contacts.**

Don Brutzman Code UW/Br Naval Postgraduate School Monterey California 93943 USA work +1.831.656.2149 fax +1.831.656.7599 brutzman@nps.navy.mil http://web.nps.navy.mil/~brutzman	Don McGregor Code MV/Mc Naval Postgraduate School Monterey California 93943 work +1.831.656.7650 fax +1.831.656.7599 mcgredo@nps.navy.mil	Alan Hudson President, Yumetech Inc. 999 Third Avenue, Suite 3800 Seattle, WA 98104-4023 USA work +1.206.340.8900 fax +1.206.328.2246 info@yumetech.com http://www.yumetech.com

Background whitepaper follows with details, continued implementation is in progress.

Cross-Format Schema Protocol (XFSP) for Binary Serialization of XML

NPS has been working for several years on a binary encoding of general XML documents and languages, intended for rapid prototyping of network protocols in support of large-scale virtual environments (LSVEs). This section summarizes the Cross-Format Schema Protocol (XFSP) technical approach, which can provide a compatible binary encoding for X3D.

Originally called Dynamic Behavior Protocol, the original approach was to define a packet-definition language in XML. Such a language might easily be used to define the packet payloads used by network protocols, then parsed by a network handler which did the necessary work of reading and writing protocols conforming to the format. This original version was implemented and working in October 2002. An important insight was then realized by Andrzej Kapolka, Ekrem Serin and Don McGregor of NPS. It turns out that XML Schema already supports most of the data-structure information needed for a packet description language. We mapped out an approach for doing so, renaming this work the Cross-Format Schema Protocol (XFSP). XFSP was implemented in November 2002 and demonstrated at the IITSEC conference in December 2002.

A simple description of the XFSP algorithm for binary serialization follows.

- XML elements are represented both by open-tag and close-tag tokens. This allows rapid regeneration of the original tree-like document structure.
- XML attributes represented by single tokens, in same namespace as elements.
- Payload data for attributes and element content is type dependent, occupies predictable lengths between tokens. Array types are preceded by field-length integers.
- Payload compression is type specific, e.g. long integers, floats, doubles, etc.
- Use of Canonical XML form (in combination with XML validation) can ensure that documents (i.e. messages) are compressed consistently, independent of arbitrary whitespace characters, apostrophe/quotation mark attribute delimitations, attribute ordering, etc. http://www.w3.org/TR/xml-exc-c14n

A valuable test of the protocol occurred when NPS masters students studying advanced XML took the IEEE DIS Specification, encoded the Entity State PDU (ESPDU) as an XML schema, and created an XFSP reader/writer all within one day. Such DIS implementation work originally took about a year. Since then, the students in the NPS MV4250 Advanced XML course have implemented approximately 3 dozen distinct PDUs. This process means that rapid generation of network protocols specifically tuned for special applications can be quickly produced merely through rigorous data-structure definition via XML schema. Similarly and more generally, any document format defined by XML schema can have a network protocol defined for it.

The basic approach for the XFSP application support proceeds as follows:

- XFSP reads an XML schema for an arbitrary tagset
- XFSP thereby produces a packet reader/writer which can handle any valid document corresponding to the XML schema
- Packets are in binary form, using tokens for elements/attributes and serialization of payload data
- Unambiguous efficient tokenization/serialization/deserialization/reconstitution

In addition to these already-significant advances, XFSP is also appropriate for production and reading of binary file formats. In some sense, this capability is obvious: a file stream can be serialized as a networked stream, and vice versa. In another sense, XFSP provides surprising new capabilities: legacy binary file formats that are otherwise inaccessible might be quickly exposed and manipulated, both for use by XML tools and as networking protocols. Such is the case even for some binary formats where such interoperability was not originally intended.

Further potential benefits for future implementations utilizing XFSP follow.

- Since the documents are in XML, they are well structured and suitable for further compression. The XMill project (online at http://www.research.att.com/sw/tools/xmill) has some interesting studies on this topic. Arbitrary documents may grow when converted to XML, but then compress better than the gzipped original.
- XML format allows simultaneous use of XML Encryption and XML Signature (i.e. Authentication), both in W3C Recommendation (i.e. approved) status. http://www.w3.org/Encryption http://www.w3.org/Signature
- XML format allows integration of other metadata, such as Resource Description Framework (RDF). http://www.w3.org/RDF
- Properly adapted, the binary encoding appears to be suitable both as a file format and also for network delivery (e.g. scene streaming, progressive rendering, event delivery, etc.)
- XFSP-derived network protocols (and corresponding binary file formats) are suitable for independent implementation in any network- or file-capable programming language.

The current version of XFSP source and final thesis describing this work [Serin 2003] are online at the open-source support site SourceForge. Detailed finite state machines (FSMs) on serialization/deserialization and extensive other details are included.

http://sourceforge.net/projects/npsnetv

Ongoing XFSP work continues to produce interesting new insights and capabilities.

[X3D ISO-19776 2003] X3D Compressed Binary Encoding Opportunities using XFSP

The Cross-Format Schema Protocol (XFSP) work provides a suitable basis for generation of a binary protocol (and hence binary file format) for X3D. The details of this approach are explored in Chapter 6 of the [Serin 2003] thesis. NPS has proposed the X3D schema serialized via XFSP as the basis for an X3D Compressed Binary Encoding.

The primary goals of the possible approaches to Compressed Binary X3d are as follows:

- smaller X3D files
- faster loading at run-time
- streaming, for incremental loading, incremental additions and also subgraph replacement
- all other points in the X3D Binary Requirements Request for Proposals (RFP) which is available online at http://www.web3d.org/TaskGroups/x3d/X3dBinaryRFP.html

Interestingly, this approach was previously deferred while other X3D requirements were met. A composite solution which simultaneously harmonizes all of these sometimes-competing requirements now appears to be technically feasible.

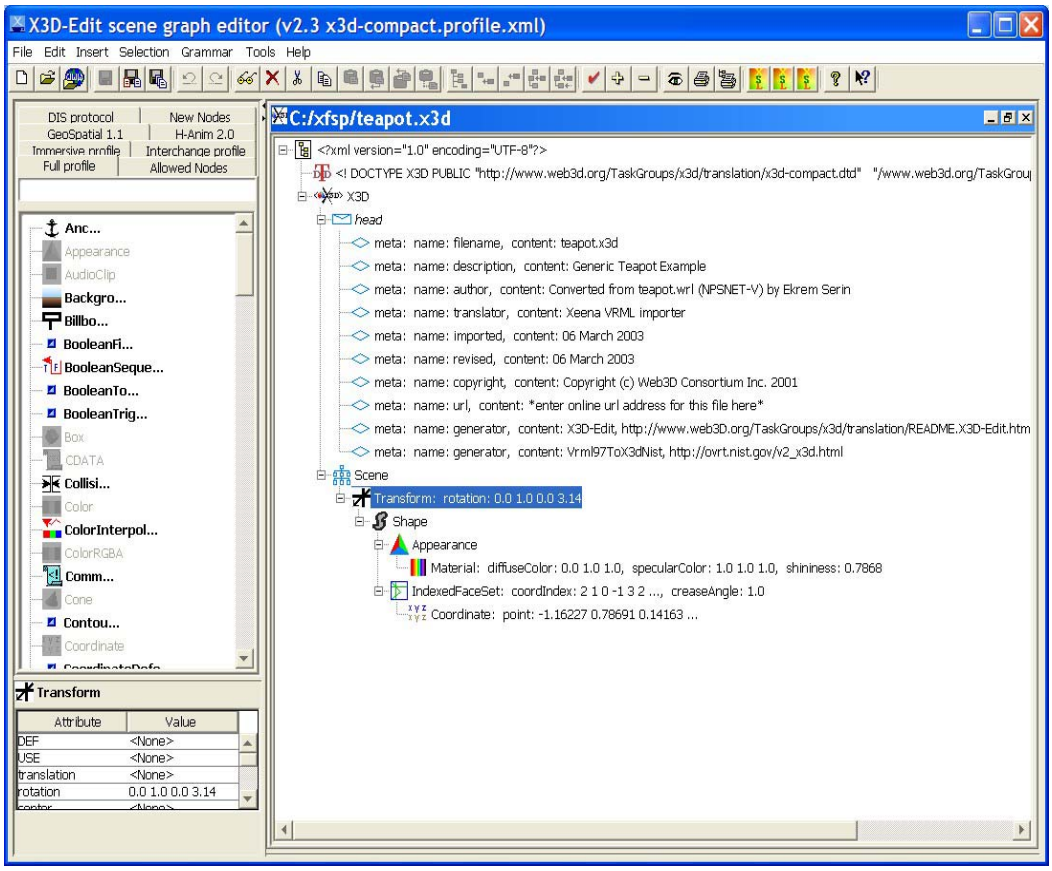
Additional considerations follow for tuning XFSP to best support an X3D binary encoding.

- XML Encryption and XML Signature provide default royalty-free (RF) algorithms but also allow specification of alternate algorithms. This technique might again be applied in an X3D binary encoding to allow both open and extensible algorithms for geometry compression, quantization, encryption etc.
- Major performance speedups are expected over gzip during binary scene loading, since the binary structures are directly ready for immediate insertion into scene-graph data structures in memory. No further character-based parsing is needed for numeric fields.
- Fixing the token set values assigned to X3D nodes and fields permits immediate streamed deserialization upon commencing loading/receipt.
- Example XFSP implementation of Binary X3D is offered using Java. Even better performance might be achieved through a tuned X3D-binary implementation, rather than an XFSP implementation capable of general XML.
- Tuned X3D-XML implementations might be adapted to handle VRML encoding directly. Alternatively, 1-1 translation to X3D's XML encoding allows corresponding usage by VRML encoding also.
- XFSP and an X3D binary encoding are offered by NPS under royalty-free terms in accordance with the Web3D Consortium Intellectual Property Rights (IPR) Policy http://www.web3d.org/aboutus/ipr.html [Web3D 2001]
- Quantization tables (i.e. indexed lookup of common/averaged values) will likely be allowed. This is particularly practical for high-resolution floating-point types that might actually consist of few discernibly different values.

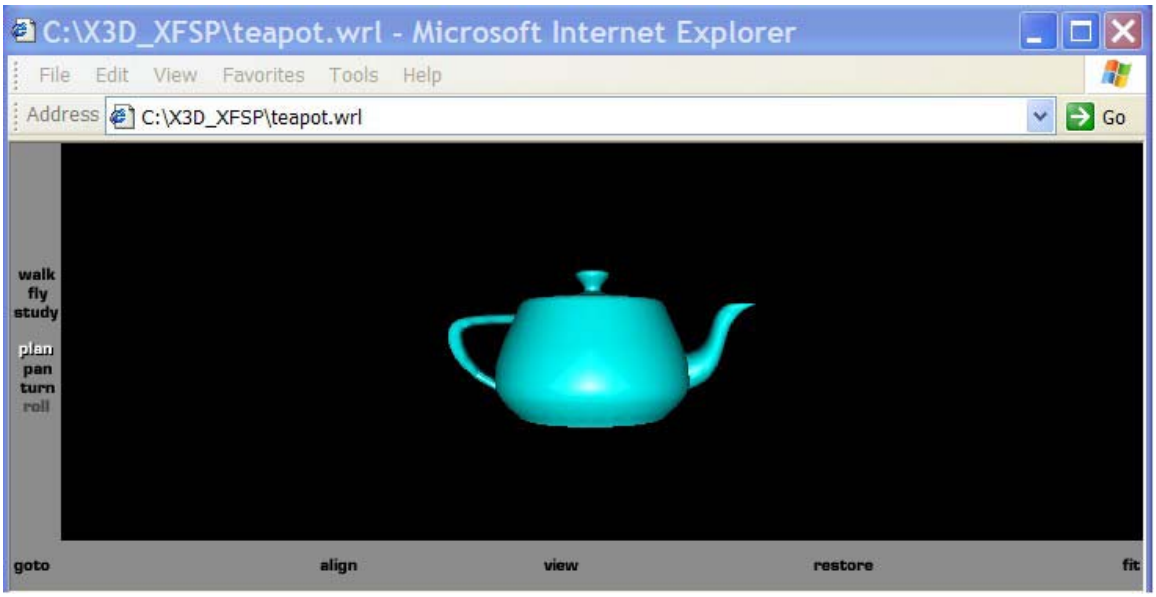
- Further typing may be applied for special geometry-related types like Color, Normal, coordinates, indices, pixelTexture definitions, enumerations, unique-identifier UUID values (e.g. for CAD3D), etc. Thus special typing (or type extensibility) might support specialty compression components on a geometric-type by geometric-type basis.
- Separate namespaces for prototype/field names and DEF/IDs.

Initial Compression Results for X3D Binary Encoding

Initial binary compression capabilities for X3D are demonstrated in the [Serin 2003] XFSP thesis. Thesis excerpts and experimental results follow.



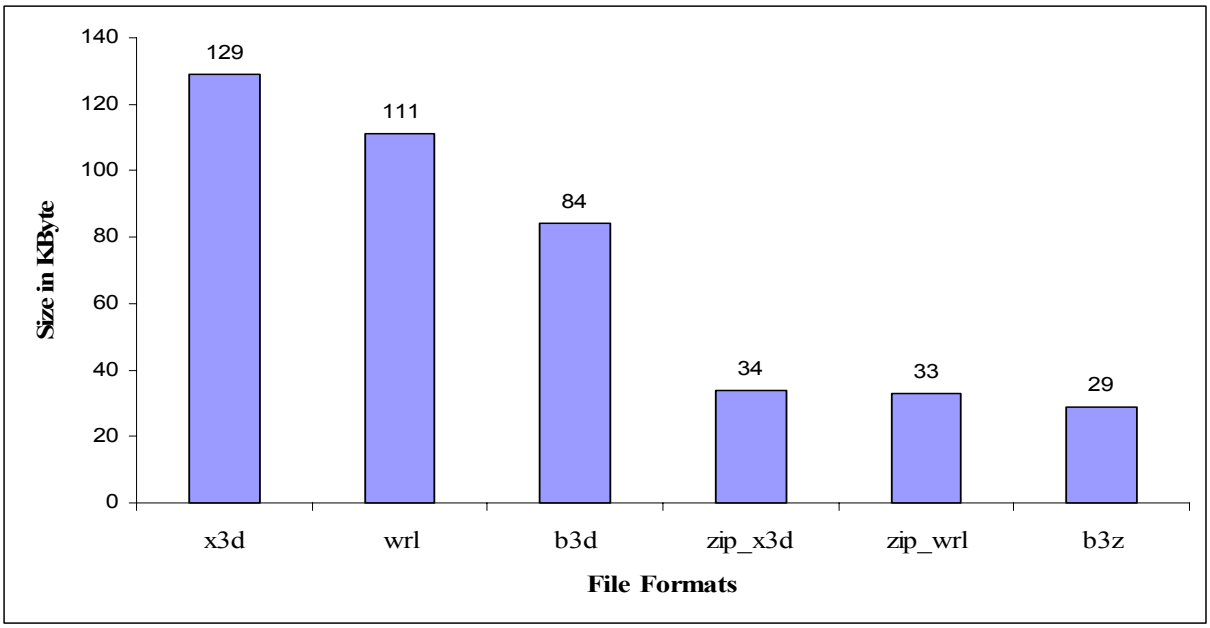
In order to effectively measure the efficiency of X3D binary serialization, a representative yet computationally demanding example was needed. Since the teapot has long been a canonical test example used in the historic development computer graphics, a teapot model was created in X3D. The following figures respectively show this scene in X3D-Edit and rendered as VRML in the *Cortona* browser [Parallel Graphics 2003]. The teapot example scene provides a geometrically demanding exemplar for binary compression.



Teapot geometry is a large single IndexedFaceSet, rendered in Cortona VRML browser [Parallel Graphics 2003].

The following uncompressed and compressed file formats are examined in [Serin 2003] to compare relative effectiveness at the initial XFSP binary-serialization. Corresponding compression results appear in the following figure.

- x3d : X3D File Format
- wrl : VRML97 File Format
- b3d : Binary X3D File Format
- zip_x3d : X3D File Compressed by WinZip Program
- zip_wrl : VRML97 File Format Compressed by WinZip Program
- b3z : X3D File Compressed by Serializer using GZIP Streams

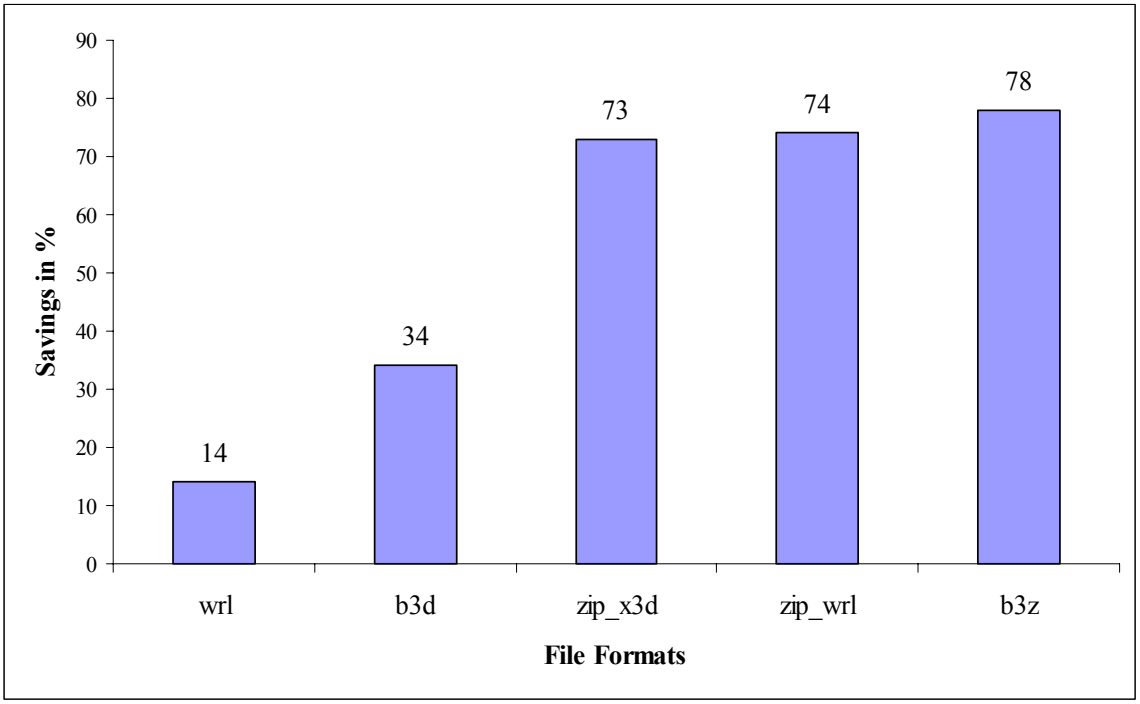


File format compression comparison for teapot exemplar [Serin 2003].

Examining the figure above shows a number of interesting results. First, XML encodings of .x3d files can often be larger than alternate text encodings such as .wrl VRML format. Second, binary serialization of node/field tokens and numeric data shows an immediate improvement over plain-text encodings. Third, the .zip reduction of .x3d and .wrl formats to a consistent size matches expectations. Zip and gzip compression algorithms look for text patterns and typically produce similar results, within 1-2%. Finally, a further 10% reduction was demonstrated when gzipping the .b3d binary serialization. This is excellent progress. Further reductions are expected with the future addition of geometric compression techniques such as polygon combination, more efficient vertex representations, quantization of color and normal values, etc.

Clearly there are significant benefits to XML compression. Further optimizations will make X3D binary compression further valuable. As an alternate comparison, recomputes the compression results of in terms of percent size savings.

In summary, many requirements and considerations exist but it appears that a composition of all declared X3D binary encoding goals might be compatibly achieved. Ongoing work continues.



Percentage compression relative to original file size [Serin 2003].

X3D Streaming Considerations

Yet another interest for X3D Graphics is the ability for a binary compressed file format to support progressive run-time streaming of already-rendered scene graphs. Live streaming considerations include:

- Need to investigate whether to solely support streaming of XML attribute (i.e. X3D field) payloads, or to also support streaming of combined XML elements (i.e. X3D nodes).
- Multiple update modes may be necessary for uncorrupted updates of field data. Three approaches are currently foreseen:
    - direct replacement
    - progressively append arrays (e.g. stream continuously added motion-capture data)
    - buffer until update is completely delivered, then swap with original
- Event delivery likely contains regular binary segment consisting of node ID, field token and payload data, along with (optionally honored) event timestamp.

  X3D streaming design and implementations remain an active area of work.